Организация баз данных
2001. №2

УДК 519.683

© 2001 г. **В.А. Анненков**,
Е.А. Нурминский, д-р физ.-мат. наук,
С.В. Смирнов, канд. физ.-мат. наук
(Институт автоматики и процессов управления ДВО РАН, Владивосток)

ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ЭВМ С ИЕРАРХИЧЕСКОЙ ОРГАНИЗАЦИЕЙ ПАМЯТИ НА ПОДПРОГРАММАХ БИБЛИОТЕКИ BLAS[1]

Рассматривается прохождение алгоритмов базовой линейной алгебры в вычислительной системе с многоуровневой иерархической организацией памяти. Изучается влияние особенностей архитектуры, страничного механизма памяти и специальных способов размещения данных на производительность вычислительного устройства.

Введение

Стремительно нарастающая массовая потребность в высокопроизводительных вычислительных ресурсах, необходимых для решения разнообразных задач [3,4], приводит к широкому использованию открытых систем массового параллелизма, состоящих из стандартных компонентов, в том числе массовых серийных микропроцессоров. Рост вычислительных возможностей современных высокопроизводительных вычислительных систем связан не только с переходом на высокочастотные элементы, но и во многом определяется интенсивным развитием средств параллельной работы в аппаратуре ЭВМ.

Увеличение производительности может достигаться с помощью самых разнообразных форм параллелизма. Любая вычислительная система представляет собой совокупность связанных функциональных устройств. В каждый момент времени эти устройства могут либо простаивать, либо выполнять полезную работу – хранение, передачу или обработку информации. Относительное быстродействие вычислительной системы опреде-

[1] Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 01-07-90225) и федеральной целевой программы «Интеграция» (код проекта В-0009).

ляется составом используемых функциональных устройств и степенью их общей загруженности.

В соответствии с этим можно выделить тенденции повышения быстродействия вычислительных систем. Одна из них связана с использованием большого числа одновременно работающих функциональных устройств и приводит к проблеме распараллеливания вычислений, другая – с увеличением эффективности использования самих устройств. Повышение загруженности функциональных устройств связывают с конвейеризацией вычислений. Максимальная загрузка вычислительного конвейера достигается при наличии большого числа длинных ветвей независимых вычислений и при невысоких накладных расходах на передачу данных при реализации этих ветвей.

Необходимость решать трудоемкие задачи заставляет добиваться эффективности процесса решения на ЭВМ и учитывать ее архитектурные особенности. При этом возникает задача отображения алгоритмов на вычислительные системы. При разработке алгоритмов для решения крупномасштабных задач, требующих обработки большого количества данных, особое внимание уделяется архитектурным особенностям основной оперативной памяти вычислительной системы, пропускной способности шины передачи данных и характеристикам доступа к разным компонентам памяти. В этой ситуации анализ передачи данных между основной оперативной памятью и функциональными устройствами становится важной частью совместного исследования расчетного алгоритма и вычислительной системы.

Достижение высокой производительности вычислительных устройств за счет программных средств возможно при использовании компиляторов, обеспечивающих высокий уровень оптимизации. Другой подход состоит в применении базовых процедур, в высокой степени влияющих на производительность широкого круга приложений. Когда для таких процедур определен интерфейс и сформулирован соответствующий стандарт, их можно оптимизировать под различные архитектуры. В линейной алгебре процедуры, содержащие алгебраические операции над матрицами и векторами, собраны в библиотеке базовых подпрограмм линейной алгебры BLAS (Basic Linear Algebra Subprograms). Документация BLAS находится по адресу http://www.netlib.org/blas.

В данной работе исследовались взаимодействие и обмен данными между функциональными устройствами и основной оперативной памятью внутри процессорного элемента комплекса МВС-1000/16. Изучение особенностей передачи данных между основной оперативной памятью и функциональными устройствами процессорного элемента необходимо для решения проблемы отображения алгоритмов на архитектуру вычислительной системы. Проведено исследование производительности процессорного элемента комплекса МВС-1000/16 на базовых операциях линейной алгебры для различных реализаций библиотеки BLAS. Исследование произво-

дительности отдельного процессорного элемента (уровень микропараллелизма) является необходимым этапом исследования общей эффективности работы параллельных вычислителей.

Вычислительный комплекс МВС-1000/16 в ИАПУ ДВО РАН

Система МВС-1000/16, установленная в ИАПУ ДВО РАН (http://www.dvo.ru/bbc/hardware/mbc1000/), представляет собой параллельный вычислитель, разработанный совместно ИПМ им.М.В.Келдыша и НПО ”Квант”.

МВС-1000/16 состоит из шестнадцати вычислительных узлов на базе микропроцессоров Intel Pentium III с частотой 800 МГц, кэш-памятью второго уровня 256Кбайт, оперативной памятью 256 Мбайт. Вычислительные узлы объединены в кластер двумя сетями Fast Ethernet. Каждый процессорный элемент системы входит в две независимо коммутируемые сети (управляющую и связную). Для этого служат два коммутатора, функционирующие в режиме 100 Мбит/с. Один узел имеет дополнительный интерфейс для выхода во внешнюю сеть. Управляющая сеть обеспечивает функции доступа процессорных элементов к общим периферийным ресурсам. Связная сеть служит для передачи сообщений внутри параллельных программ, работающих на группе процессорных элементов.

Отображение алгоритмов на вычислительные системы

Будем считать, что алгоритм или описывающее его правило позволяют каким-либо образом определить: множество переменных, в преобразовании которых заключается реализация алгоритма, множество операций, выполняемых в процессе реализации алгоритма, соответствие, показывающее, какие результаты выполнения предшествующих операций являются аргументами для каждой операции. Множеству операций алгоритма поставим во взаимно однозначное соответствие некоторое множество точек. Если аргумент одной операции есть результат выполнения другой операции, то соответствующие точки соединим дугой. Построенный таким образом граф и будем называть графом алгоритма. Задача отображения алгоритмов на вычислительные системы формулируется следующим образом [1]: дано описание вычислительного алгоритма в виде графа и указана спецификация его операций, дано описание вычислительной системы и указана спецификация ее устройств. Необходимо ответить на вопросы – можно ли реализовать алгоритм на данной системе и в случае положительного ответа указать реализацию алгоритма за минимальное время?

Граф алгоритма отражает то существенное в связях между операциями, что и хотел показать разработчик алгоритма. При рассмотрении графа появляется возможность исследовать всю совокупность реализаций алгоритма, эквивалентных с точки зрения связей между операциями. На

основе такого исследования можно выбрать реализацию, подходящую для конкретной вычислительной системы. При отождествлении алгоритма с графом предполагается, что в графе уже отражены все объекты и связи, влияние которых на реализацию алгоритма подлежит изучению. В частности, если требуется учесть временные затраты на передачу информации между функциональными устройствами и основной оперативной памятью, то в граф должны быть включены соответствующие вершины и дуги, отражающие операции такого типа [2].

Решение задачи отображения в общей постановке требует неоправданно больших затрат, чрезмерно усложняет и запутывает изложение алгоритма. Нецелесообразно выбирать в качестве нижнего уровня операторы машинных команд, т.е. доводить изложение до конкретных функциональных устройств. В качестве нижнего уровня детализации алгоритма пользователь системы может выбрать, например, хорошо оптимизированные процедуры базовых операций линейной алгебры, собранные в пакете BLAS. В этом случае задача отображения может отдельно решаться в процессе разработки базовых библиотек программного обеспечения (ПО), максимальным образом учитывающих архитектуру процессорного элемента вычислительной системы. Одним из эффективных подходов к разработке ПО является AEOS – Автоматическая эмпирическая оптимизация программного обеспечения (Automated Empirical Optimization of Software).

Примером реализации идеологии AEOS в линейной алгебре является проект ATLAS – Автоматически настраиваемое программное обеспечение линейной алгебры (Automatically Tuned Linear Algebra Software). ATLAS является свободно распространяемой, многоплатформенной, адаптирующейся под архитектуру конкретного вычислительного устройства библиотекой. В ATLAS оптимальным образом реализованы процедуры BLAS. Автоматическая настройка библиотеки ATLAS осуществляется методом эвристического поиска. Учитывается иерархическая организация памяти, размер компонентов кэш-памяти, глубина конвейера и другие детали архитектуры. Итогом является оптимальная настройка процедур BLAS под конкретную вычислительную платформу [5].

Результаты вычислительных экспериментов

При анализе реальной производительности процессорного элемента необходимо рассматривать все возможные варианты взаимного расположения данных. Это обусловлено архитектурными особенностями вычислителя. Максимальные показатели производительности достигаются при расположении данных во внутрикристальном кэше L1. Высокие показатели производительности наблюдаются также и при расположении данных во вторичном кэше L2. Размер кэша накладывает ограничения на длину векторов, которые могут в нем целиком поместиться. Такие векторы далее

будем называть короткими. Соответственно длинными векторами назовем те, которые не помещаются целиком в кэш-память. Скорость обмена данными по внешней системной шине существенно ниже скорости обмена данными между процессором и кэш-памятью. Это приводит к низкой загрузке процессора в случае, когда данные располагаются в основной оперативной памяти.

Для проведения исследований был разработан комплекс программ на языке FORTRAN-90. Реализовано таймирование различных вариантов взаимного расположения данных. Изучение особенностей взаимодействия между функциональными устройствами и основной оперативной памятью приводит к необходимости рассматривать различные способы взаимного расположения данных внутри процессорного элемента.

Для анализа реальной производительности рассматривались случаи расположения данных целиком в кэш-памяти, целиком в основной оперативной памяти и промежуточный вариант, когда часть данных располагается в кэш-памяти, а часть поступает по внешней системной шине из основной оперативной памяти. Таймирование вычислений осуществлялось с помощью высокоточной процедуры MPI_WTIME из пакета MPICH-1.2.1.

Для проведения экспериментов использовалась библиотека ATLAS-3.2.1, библиотека BLAS, поставляемая с оценочной версией компилятора PGI Workstation-3.1 и исходные тексты процедур BLAS, доступные в Интернете по адресу http://www.netlib.org/blas. Библиотека BLAS состоит из процедур векторно-векторных операций (первый уровень), матрично-векторных операций (второй уровень), матрично-матричных операций (третий уровень). На каждом уровне были выделены наиболее характерные процедуры, которые использовались в исследованиях. Все вычисления производились с двойной точностью (размер слова – 8 байт).

Опишем наиболее характерные черты поведения производительности по мере роста объемов обрабатываемых данных. Наблюдается первоначальный быстрый рост производительности при малых длинах векторов. Это обусловлено уменьшением доли накладных расходов на организацию вычислений. Высокие показатели производительности отмечаются при расположении данных в кэш-памяти.

Для больших объемов обрабатываемых данных производительность процессорного элемента определяется конструктивными особенностями системной шины и страничным механизмом организации памяти. Использование аппаратных особенностей работы с кэш-памятью позволяет добиться увеличения производительности за счет специальных способов размещения данных.

Наиболее характерные графики результатов представлены на рис.1-4 (по горизонтали масштаб логарифмический).

Для процедур первого уровня наблюдается наиболее выраженная зависимость производительности от фактического месторасположения дан-

ных. Производительность вычислений с короткими векторами, расположенными в кэш-памяти, может на порядок превышать производительность вычислений с длинными и с короткими векторами, расположенными в основной оперативной памяти.

Проиллюстрируем это на примере операции DAXPY (см. рис.1). Кривая а) соответствует случаю, когда короткие векторы расположены в кэш-памяти. Кривые б) и в) отражают случай, когда в диапазоне коротких векторов один из аргументов берется из кэш-памяти, другой – из оперативной памяти. Кривая г) соответствует случаю, когда оба аргумента берутся из основной оперативной памяти. При увеличении длины векторов происходит исчерпывание кэш-памяти для случаев а), б), в), и все кривые выходят на один и тот же режим, когда производительность определяется только характеристиками внешней системной шины.

В процедурах второго уровня компоненты вектора многократно используются и автоматически попадают в кэш-память, поэтому производительность зависит от фактического расположения матрицы. Наибольшая производительность достигается в случае расположения матрицы целиком в кэш-памяти.

Рассмотрим операцию DTRSV (см. рис.2). На кривой а) представлен случай, соответствующий обращению нижней треугольной матрицы. Кривые б) и в) представляют случаи обращения верхней треугольной исходной и транспонированной матриц соответственно. Наблюдается резкое падение производительности, когда матрица уже не помещается в кэш-память. После исчерпывания кэш-памяти основное влияние на производительность оказывают особенности реализации механизма выборки данных из основной оперативной памяти. Например, двукратное снижение производительности в случае в) по сравнению со случаем а) объясняется разницей в порядке выборки данных из основной оперативной памяти (по убыванию и по нарастанию адресов).

Сравнительный анализ результатов таймирования прохождения процедур третьего уровня из различных реализаций библиотеки BLAS показывает преимущества пакета ATLAS. В нем используется алгоритм разбиения матриц на блоки, при котором максимально возможное количество данных, над которыми производятся действия, находится в кэш-памяти. Этим и объясняется резкое увеличение производительности.

На рис.3 представлен результат прохождения подпрограмм, реализующих операцию умножения матриц общего вида DGEMM. Использовались подпрограммы из библиотеки ATLAS (кривая а) и библиотеки BLAS, поставляемой с PGI Workstation 3.1. (кривая б). В данном примере при применении библиотеки ATLAS наблюдается десятикратный выигрыш в производительности.

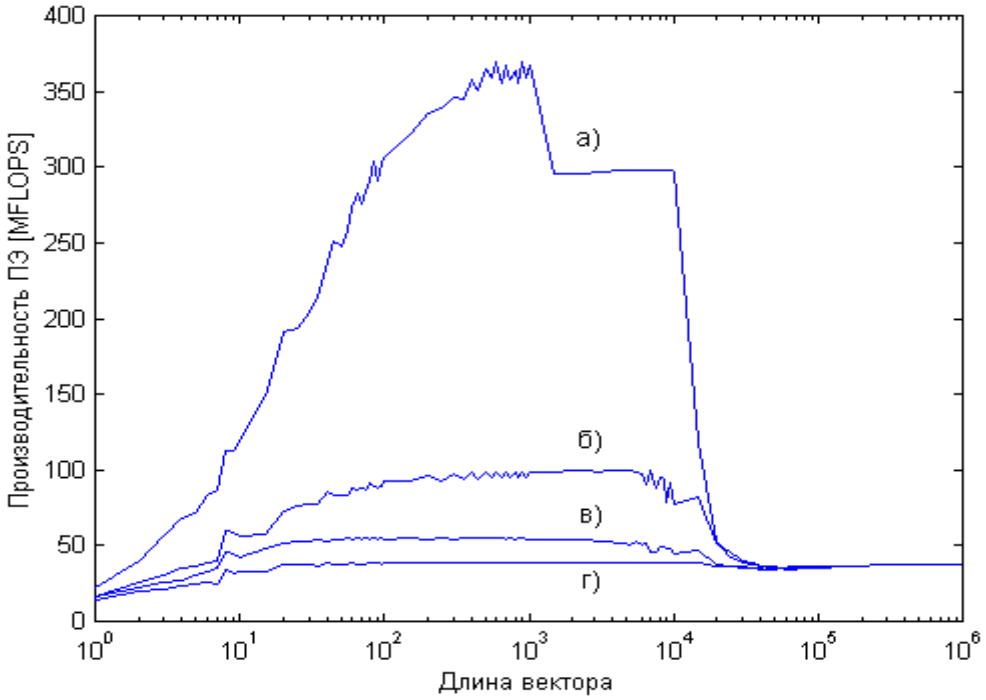

Рис.1. Производительность на операции первого уровня DAXPY

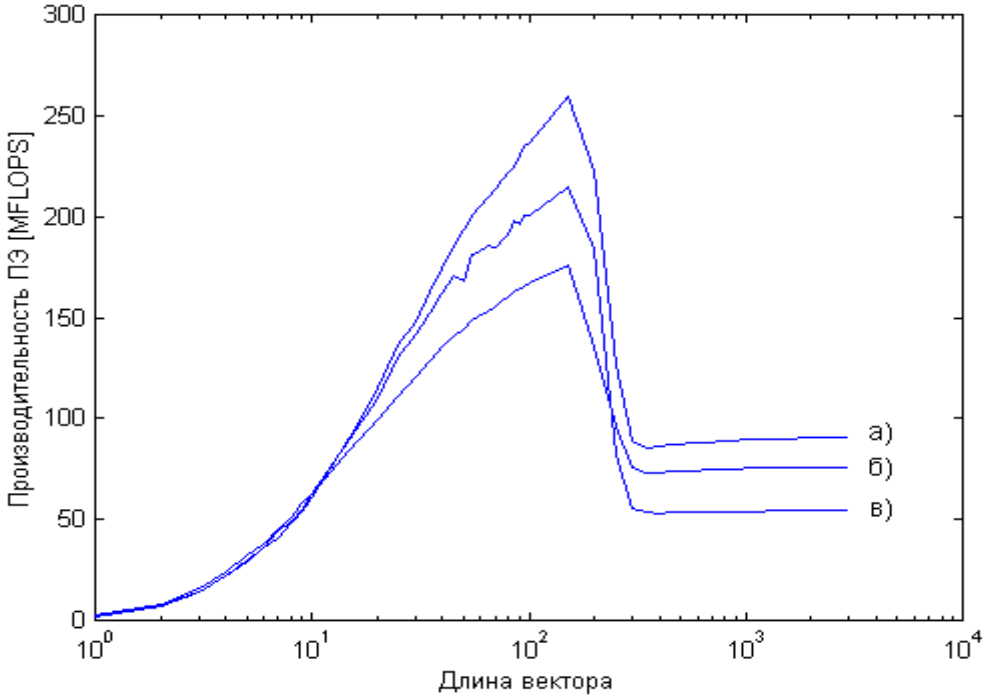

Рис.2. Производительность на операции второго уровня DTRSV

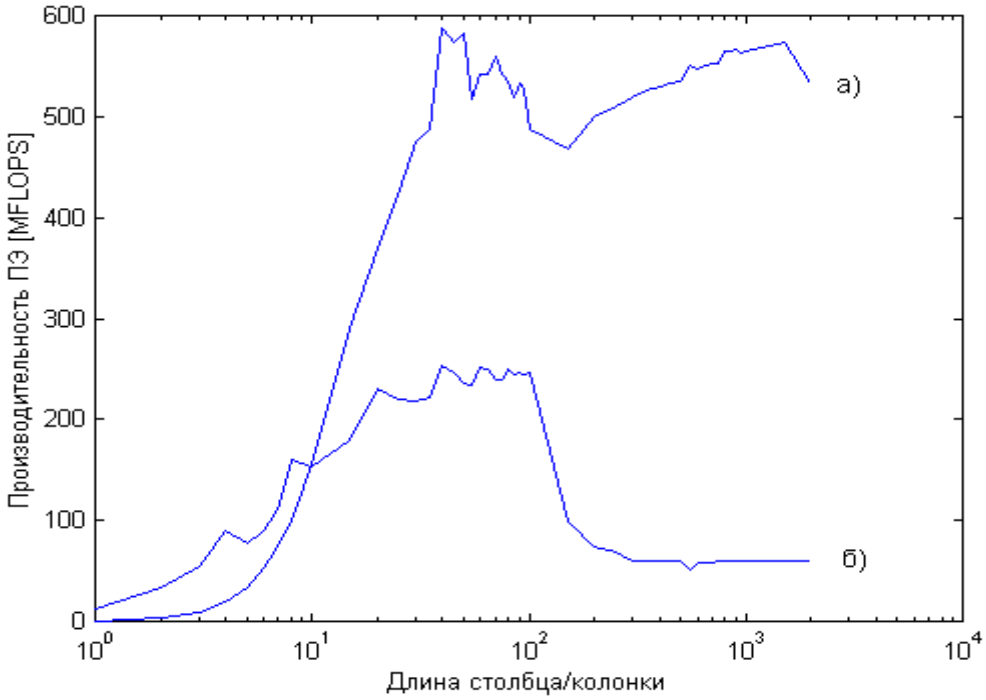

Рис.3. Производительность на операции третьего уровня DGEMM

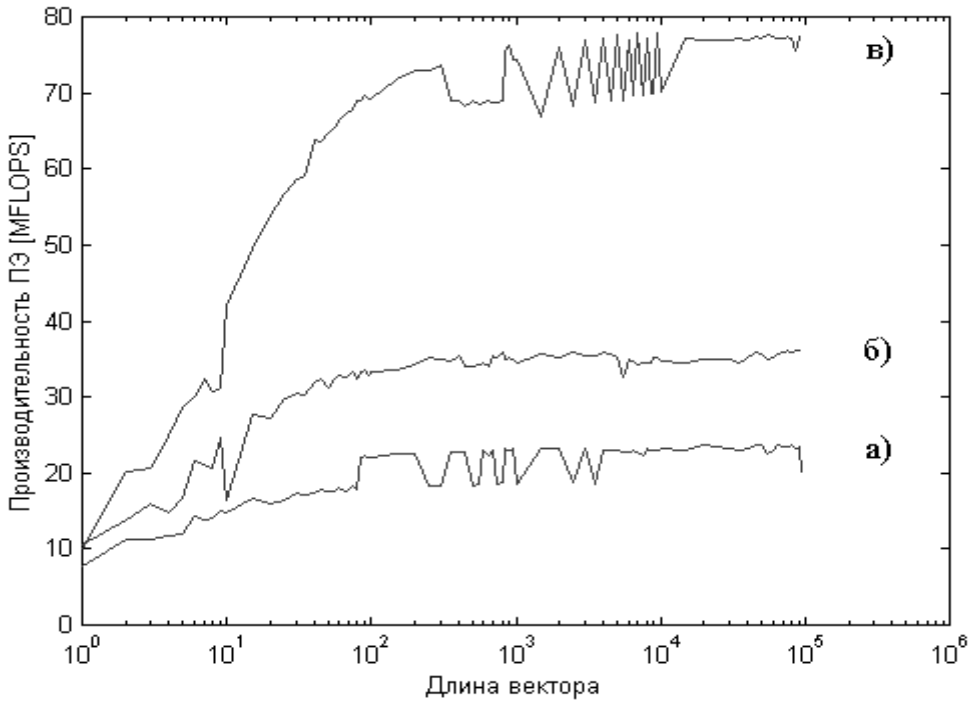

Рис.4. Производительность на операции первого уровня DDOT

В рассмотренных выше примерах элементы векторов и матриц были размещены в памяти по смежным адресам. В случае специального размещения данных, когда, например, элементы вектора расположены через равные промежутки, следует принимать во внимание особенности аппаратной реализации работы с кэш-памятью. Копирование данных из оперативной памяти в кэш производится сегментами (линиями) размером 32

байта. Если промежутки между элементами векторов настолько большие, что из скопированных 32 байтов используется только 8, это может привести к четырехкратному замедлению вычислений по сравнению со случаем, когда элементы векторов расположены без промежутков. На рис.4 представлены показатели производительности для случаев специального размещения данных на примере операции первого уровня DDOT. Все векторы были расположены в оперативной памяти. Кривая а) соответствует размещению элементов векторов с промежутком, равным 24 байтам (используется только одно слово из линии), кривая б) соответствует размещению с промежутком, равным 8 байтам (используются уже два слова из линии), кривая в) соответствует расположению элементов по смежным адресам (используются все четыре слова из линии). Как видно из графиков, производительность в значительной степени определяется эффективностью использования внешней шины данных.

Заключение

В данном исследовании изучались особенности обмена данными между функциональными устройствами и основной оперативной памятью внутри процессорного элемента многопроцессорного вычислительного комплекса МВС-1000/16. Исследуемый процессорный элемент представляет собой систему с многоуровневой иерархической организацией памяти. Рассматривалось прохождение базовых процедур линейной алгебры на примере библиотеки BLAS. Выявлены основные факторы влияния иерархической организации и страничного механизма оперативной памяти на производительность процессорного элемента. Также изучалось влияние на производительность процессорного элемента специальных способов размещения данных. Результаты сравнения показывают, что для базовых операций линейной алгебры наиболее целесообразным представляется применение пакета ATLAS, в котором реализован механизм эффективного использования многоуровневой иерархической организации памяти, что приводит к многократному увеличению производительности для процедур третьего уровня. Отметим, что для первого и второго уровней BLAS использование библиотеки ATLAS не приводит к столь существенному увеличению производительности. Для эффективного использования процедур BLAS первого уровня в общем случае необходимо оптимизировать алгоритм задачи, чтобы вычисления велись преимущественно с векторами, расположенными в кэш-памяти. Для широкого круга задач с большим количеством данных, когда невозможно эффективно задействовать кэш-память, для оптимизации вычислительного процесса необходимо учитывать особенности реализации механизма выборки данных из основной оперативной памяти. За счет учета аппаратных особенностей работы с кэш-памятью и специальных способов размещения данных можно добить-

ся увеличения производительности при осуществлении векторных операций с данными, расположенными в памяти через равные промежутки.

ЛИТЕРАТУРА

1. *Воеводин В.В.* Математические модели и методы в параллельных процессах. М.: Наука, 1986.
2. *Воеводин В.В.* Параллельные структуры алгоритмов и программ. М., 1987.
3. *Корнеев В.В.* Параллельные вычислительные системы. М., 1999.
4. *Zabrodin A.V., Levin V.K., Korneev V.V.* The massively parallel computer system MBC-100. Lecture Notes in Computer Science. № 964 // Parallel Computing Technologies. Third International Conference, PaCT-95. St.Petersburg: Springer. 1995. P.341-355.
5. *Whaley R.C., Petitet A., Dongarra J.J.* Automated Empirical Optimization of Software and the ATLAS project. WWW.netlib.org/atlas.

УДК 519.68

© 2001 г. **А.В. Бурдаков**,
Ю.А. Григорьев, д-р техн. наук
(Московский государственный технический университет им. Н.Э. Баумана),
А.Д. Плутенко, канд. техн. наук
(Амурский государственный университет, Благовещенск)

ОЦЕНКА ВРЕМЕНИ ВЫПОЛНЕНИЯ ЗАПРОСОВ К ОБЪЕКТНО-ОРИЕНТИРОВАННЫМ БАЗАМ ДАННЫХ

Предложен математический метод оценки времени выполнения запросов к объектно-ориентированным базам данных. Рассмотрены алгоритмы выполнения запросов Forward Join и Reverse Join.

Введение

Разработка распределенных систем обработки данных (РСОД) сопряжена с оценкой различных функциональных и нефункциональных показателей предлагаемых решений. Одними из важных характеристик являются индексы производительности РСОД. Для оценки этих показателей на ранних этапах разработки системы необходимо применять специальные